Adaptive Time-based Encoding for Energy-Efficient Large Cache Architectures

Payman Behnam
University of Utah
Salt Lake City, UT
behnam@cs.utah.edu

Naser Sedaghati
Imagination Technologies
San Francisco, CA
naser@sedaghati.org

Mahdi Nazm Bojnordi
University of Utah
Salt Lake City, UT
bojnordi@cs.utah.edu

ABSTRACT

Demanding larger memory footprint and relying heavily on data locality has made last-level cache (LLC) a major contributor to overall energy consumption in modern computer systems. As a result, numerous techniques have been proposed to reduce power dissipation in LLCs via low power interconnects, energy-efficient signaling, and power-aware data encoding. One such technique that has proven successful at lowering dynamic power in cache interconnects is time-based data encoding that represents data with the time elapsed between subsequent pulses on a wire. Regrettably, a time-based data representation induces excessive transmission delay per every block transfer, thereby degrading the energy efficiency of memory intensive applications.

This paper presents a novel adaptive mechanism that monitors characteristics of every application at runtime and intelligently uses time-based codes for LLC interconnects, thereby alleviating the diverse impact of longer transmission delay in time-based codes while still saving significant energy. Two adaptation approaches are realized for the proposed mechanism to monitor 1) application phases and 2) memory bursts. Experimental results on a set of 12 memory intensive parallel applications on a quad-core system indicate that the proposed encoding mechanism can improve system performance by an average of 9%, which results in improving the system energy-efficiency by 7% on average. Moreover, the proposed hardware controller consumes less than 1% area of a 4MB LLC.

CCS CONCEPTS

• **Computer systems organization** → **Architectures**; • **Hardware** → *Integrated circuits*;

KEYWORDS

Energy-efficient architectures, time-based encoding

ACM Reference Format:
Payman Behnam, Naser Sedaghati, and Mahdi Nazm Bojnordi. 2017. Adaptive Time-based Encoding for Energy-Efficient Large Cache Architectures. In *E2SC'17: E2SC'17: Energy Efficient Supercomputing, November 12–17, 2017, Denver, CO, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3149412.3149417

1 INTRODUCTION

Contemporary microprocessors, optimized for operation latency, tend to invest majority of the real estate accommodating very large on-chip caches in order to achieve faster data transfer. Last-level caches (LLCs) alone can consume a significant fraction of the total processor energy. In large on-chip caches, transferring data blocks between the cache controller and internal data arrays leads to significant switching activities over long and capacitive wires that dissipate more than 80% of the cache energy [24]. The latest scaling trends also approve that such high level of dynamic power for interconnects will continue in the billion-transistor processors era [8, 16].

Architectural techniques have been developed to improve dynamic energy consumption of the LLC interconnect, focusing on different aspects of the equation $P = \alpha C V^2 f$, where *alpha* represents switching activity, C is the capacitance load, V is the supply voltage, and f is frequency. However, techniques for reducing the activity factor on wires have proven to be more applicable to different classes of interconnects [4, 7, 32, 37]. Besides data compression mechanisms [9, 10, 26, 29, 33], data encoding is also considered as an effective approach in reducing the number of state transitions (i.e. bit flips) on a wire.

A recent data encoding scheme, DESC [6], showcased time-based data representation [27] for reducing the activity factor on the highly capacitive LLC interconnects. As opposed to standard binary encoding where every clock cycle on a bus wire could represent a single bit (i.e. and potentially a state transition), DESC represents information based on the time (i.e. number of cycles) elapsed between two consecutive pulses on a single wire; limiting the number of switching events to one per transmission. However, such low level of switching activities could come at a high price of sensitivity to the transferred values.

For illustration purposes only, consider transferring a block of B bits over W wires. Standard binary encoding can work with $W_B = W$ wires, while DESC only requires $W_D = \frac{W}{C}$ where C indicates the *chunk size.* In this setting, standard binary consumes $T_B = \lceil B/W_B \rceil$ cycles (in parallel mode) whereas DESC can finish the transfer in x cycles where $x \in [\beta, 2^C \beta]$ and $\beta = \frac{B}{C \times W_D}$. For example, for a 512-bit block on a 256-wide bus, standard binary signaling takes exactly 2 cycles while DESC (with $C = 4$) can instead consume from 2 to 30 cycles (on a narrower $W_D = 64$ bit bus). Consequently, DESC's transfer latency becomes very sensitive to the value range of x, making it not always the best choice from performance point of view.

Figure 1 shows the average time consumed for transferring cache blocks for a set of 12 parallel applications on a 4MB last level cache. [1]

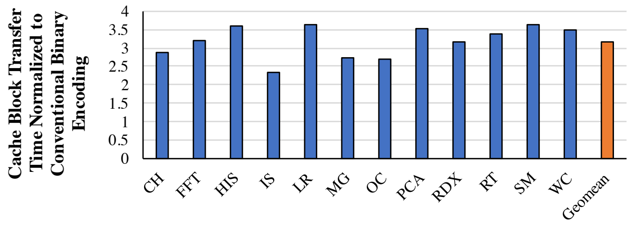

Figure 1: DESC bandwidth overheads: average number of cycles consumed for transferring cache blocks over a 4MB last level cache.

Figure 1 confirms that, on average, a data transfer using DESC interface could take 3.15× longer than the standard binary counterpart. Although DESC has shown overall energy improvements because of reduced switching activity and fewer wires [6], this increase in bandwidth (i.e. by 3.15×) has a significant impact on the overall execution time and system energy. Such combined effect on performance and energy signifies that in order to have an energy-efficient LLC interconnect, the system needs to be adaptive and able to optimize for both (i.e. using metrics such as energy-delay product *EDP*).

This paper presents an adaptive time-based encoding scheme that can learn the execution phases and memory access bursts in the application to make intelligent adjustments in the data exchange protocol accordingly. The key idea is to accurately detect phases and bursts of LLC requests and provide an encoding choice that can be changed at run-time depending on the workload behavior. Two approaches are studied for such runtime adaptation: 1) a phase adaptive mechanism that relies on a history based decision making over the intervals of memory accesses, and 2) a burst adaptive microarchitecture that constantly monitors the temporal bandwidth utilization in LLC and chooses between time-based and non-time based encodings. The goal for both approaches is to optimize the overall system energy-delay product. Experimental results on a set of memory intensive applications on a four-core system show that the proposed adaptive mechanisms can improve (on average) overall system performance and energy efficiency by 9% and 7%, respectively. Moreover, hardware implementation overhead of the proposed architectures is limited to less than 1% of the available real estate of a 4MB LLC.

2 BACKGROUND

Energy consumption in traditional binary exchange methods (e.g. serial or parallel) is data dependent: the transferred data determines state transitions on the wires that directly translates into dynamic power consumption. In the case of parallel data transfer (i.e. which is mostly common in contemporary high performance memory architectures), a data block of B bits is sent over a B-wide data interconnect in a single cycle and little synchronization is needed to be done between transmitter and receiver entities. However, from power consumption point of view, and depending on the binary encoding scheme, a transfer of 0 or 1 on every wire could cause a state transition. The number of transitions (i.e. switching activity factor) ranges from 0 to B, making the total number of transitions variable at run-time. Numerous signaling schemes exist for representing 0s and 1s on the interconnect wires. Most on-chip signaling techniques employ two voltage levels for representing bits; for example, non-return-to-zero (NRZ) represents 0s and 1s with low and high voltages, and non-return-to-zero-inverted (NRZI) technique relies on transition between low and high voltages for signaling 1s and the absence transitions for showing 0s. None of these schemes are able to make the number of switching activities constant (and independent of the bit content).

A successful attempt to make the wire-switching constant, time-based data exchange protocols (e.g. Pulse-Position Modulation) are proposed wherein a data block is encoded by transmitting a single pulse in a certain point in time (as opposed to the bit values themselves). The position of that pulse in a time span translates into a desired data [14]. Although it requires a mechanism to synchronize transmitters and receivers in order to avoid potential synchronization errors, this technique has low interference and low power usage making it attractive to use in various fields. Bojnordi et al. [6] first proposed DESC[2], an energy efficient time-based encoding mechanism for LLCs to exchange data using synchronized counters. DESC partitions a block of data (i.e. cache line) into small chunks and sent each chunk over two physical wires. To specify the beginning of a transmission, DESC also uses an additional reset wire that is shared between all data lines. Depending on the value of each chunk, the corresponding wire is toggled at a certain time representing that value. At the receiving end, the number of elapsed clock cycles between the reset signal and the toggling action is computed to represent the value of the corresponding chunk. DESC-enabled LLC benefits from a reduced activity factor but at the expense of additional bandwidth. This extra bandwidth is sometimes high enough to overshadow the energy saved by reducing the activity factor.

This paper argues that blind deployment of binary (i.e. NRZ or NRZI) or DESC encoding schemes can hurt the overall system performance or energy. We present a hybrid approach in which the system is able to understand behavior of the running workload and decides when a DESC or binary encoding is beneficial to the overall system energy-delay product. Using such runtime feedback, our proposed time-based adaptive encoding (ATE) solution then take actions, accordingly.

3 ADAPTIVE TIME-BASED ENCODING

In order to address the shortcomings of both time-based encoding (e.g. longer transfer latency in DESC [6]) and standard binary signaling (i.e. high switching rate of NRZ(I)), this paper presents a novel mechanism for Adaptive Time-based Encoding (ATE) that optimizes system energy efficiency (i.e. energy-delay product *EDP*). The key idea is to provide an opportunity for the LLC interconnect to monitor the application behavior and pro-actively select "the best" encoding scheme. In other words, ATE is designed to enable adaptive deployment of time-based encoding (i.e. DESC) and guarantee higher system-level energy efficiency. From the LLC point

[1]Detailed experimental setup is provided in Section 4.

[2]DESC: data exchange using synchronized counters.

of view, the proposed scheme analyzes the application behavior based on either a fixed number of consecutive requests (i.e. a *phase*) or high-bandwidth short stream of requests (i.e. a *burst*). Accordingly, this paper presents two alternative designs: Phase-based ATE (*PATE*) and Burst-based ATE (*BATE*). This section provides thorough explanation of the design ideas and challenges for *PATE* and *BATE*.

3.1 Adaptation to Consumed Bandwidth

The proposed adaptive time-based encoding system constantly monitors the average consumed LLC bandwidth (x) in order to intelligently select the best encoding scheme to be applied to the next single (or a group of) LLC data transfer(s). In this system, there are only two choices at any point in time and the frequency of such decision depends on the frequency of monitored events (i.e. *phase* or *burst*). We define a decision function $f(x,n)$ (Equation 1) where $f \in [0,1]$ and n determines slope of transition between the two choices.

$$f(x,n) = \begin{cases} 1 & : x < \beta \\ 0 & : x > \alpha \\ (\frac{\alpha - x}{\alpha - \beta})^n & : x \in [\beta, \alpha] \end{cases} \tag{1}$$

Evaluation of the decision function relies on computing boundaries of consumed bandwidth using behavior of both NRZ(I) and DESC encoding schemes. On the one hand, the function requires average delay per transferred cache block $\beta = B/W$, where B is the LLC cache block size and W represents the number of wires. On the other hand, the function computes maximum latency of DESC encoding $\alpha = \frac{\beta(2^C-1)}{C}$, where C describes the logical chunk size of each cache block. Given the consumed bandwidth x, Equation 1 suggests three possible regions to make a decision for the best encoding scheme (Figure 2).

(1) **DESC Region** corresponds to an interval wherein average transfer latency of a set of cache blocks is less than the average latency of NRZ(I). In other words, sending previous cache block(s) haven't taken longer than what would have been, had NRZ(I) been selected for data encoding. Thus, the decision for this region is do DESC encoding for "every" transfer since history shows DESC has met the latency deadline as well as the energy consumption.
(2) **NRZ(I) Region** is when the average latency is grater than the maximum latency of DESC encoding, the function decides to go with NRZ(I) for "every" transfer since the system cannot tolerate additional latency that would have been enforced had DESC been chosen as the encoding scheme.
(3) **Proportional Region** covers a region where neither DESC or NRZ(I) can satisfy the energy needs and the transfers have to be *proportionally* divided between the two. The function value in this transitional region determines how portions of DESC or NRZ(I) are calculated. However, *slope* of the curve shows how fast DESC proportions can be turned into standard NRZ(I).

For the *Proportional Region*, transitioning from 1 to 0 can be linear (i.e. $n = 1$), sub-linear (i.e. $n > 1$), or super-linear ($n < 1$). It determines how fast the adaptive system must react to changes in the consumed bandwidth and is chosen to satisfy the need for energy efficient execution. Figure 2 depicts the linear, super-linear and sub-linear transitions for the decision functions when $n = 0.5, 1, 2$. Next, we propose two alternatives for adaptivity in data encoding and how the decision function is used.

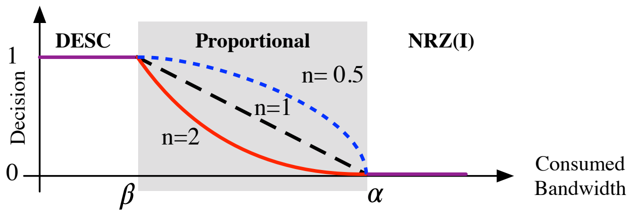

Figure 2: Adaptation to consumed LLC bandwidth.

3.2 PATE: Phase Adaptive Time-based Encoding

In order to provide adaptivity and receive constructive feedback from the system, and from LLC's point of view, a workload execution can be represented as a stream of requests $S = R_1, R_2, R_3, ..., R_s$ where each request R_i leads to a transfer of one cache block. Alternatively, stream S can be viewed as sequence of fixed-sized intervals $S = P_1, P_2, ..., P_k$, where each P_i is named a *phase* with $L = size(P_i)$ requests that are spread across time. The workload then can be divided into total of k *phases* where $k = \lceil \frac{s}{L} \rceil$. Key idea behind the proposed Phase Adaptive Time-based Encoding (PATE) is that the system should be able to *learn* from P_i and *decide* for P_{i+1} such that the overall system level energy efficiency is improved. The assumption being that majority of workloads inherently are susceptible to repeating certain patterns of memory activities (e.g. a program with regular control structures, such as for-loops, accessing predictable set of addresses). In a nutshell, and after phase P_i is over, PATE receives the feedback and updates its decision function (i.e. of choosing the DESC/NRZ(I) encoding permutations) accordingly, for the next phase P_{i+1}.

3.2.1 Computing Average Consumed Bandwidth. Figure 3 shows our proposed architecture for hardware implementation of PATE. Suppose a phase consists of L transferred blocks ($L = size(P_i)$) and

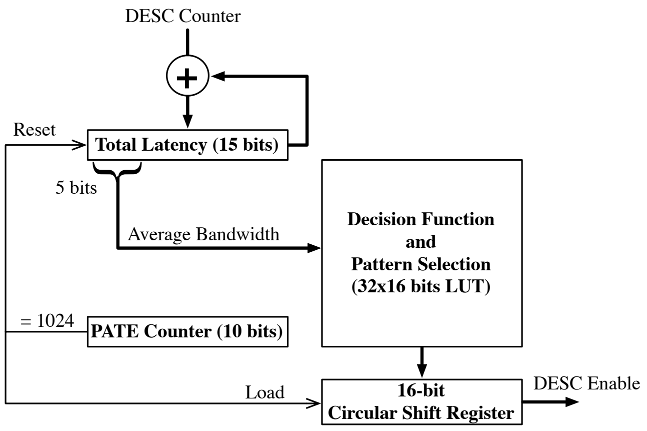

Figure 3: Illustrative example of the proposed architecture for phase adaptive time-based encoding (PATE with interval size $L = 1024$ and chunk size $C = 4$).

Table 1: Possible values for 16-bit decision patterns.

16-bit Pattern			
Binary Representation	Hexadecimal	Max 1-Dist	1-Count
0000 0000 0000 0000	0x0000	NA	0
0000 0000 0000 0001	0x0001	15	1
0000 0001 0000 0001	0x0101	7	2
0000 1000 0010 0001	0x0821	4	3
...	...	...	...
1111 1111 1111 1111	0xFFFF	0	16

chunk size of C bits. In order to indicate end of a phase, an up-counter is incremented until it reaches L, then it is reset to 0. The first interval P_0 is transferred using DESC as the selected default encoding scheme for all the L blocks. As a result, the total transfer latency T_{tot} is bounded by $T_{lb} \leq T_{tot} \leq T_{ub}$ where $T_{lb} = L\beta$ and $T_{ub} = L\beta(2^C - 1)$. In other words, $\beta \leq T_{avg} \leq \beta(2^C - 1)$. PATE computes a *truncated* average latency (T_{avg} which is the consumed bandwidth x) by dropping $\log_2(L)$ least significant bits of the total latency and using the remaining $(log_2(\beta) + C)$ MSB bits. Therefore, the "Total Latency" register is $(\log_2(L) + \log_2(\beta) + C)$-bit wide. In the illustrative example shown in Figure 3, in order to compute the average consumed bandwidth x, PATE reads only 5 most-significant bits of the total latency (and drops 10 remaining LSB bits), which helps reduce the hardware complexity as well (i.e. by avoiding floating point division for averaging).

3.2.2 Evaluation of Decision Function. To simplify the hardware even further, the decision function is implemented as a lookup table of size 2^C entries where each entry in the table represents one discrete (and approximate) value of the decision function. Each distinct value corresponds to a pre-computed bit pattern of E bits. As a result, the lookup table stores 2^C rows of E bits each. Upon reaching the end of phase P_i (i.e. PATE up-counter triggered), the computed average bandwidth x is used to find the appropriate bit pattern for the next phase P_{i+1}. The newly found pattern is then loaded into a circular right shift register which is used to apply DESC enable/disable decisions for the next L transfers.

Table 1 shows how $f(x, n)$ is converted into a set of pre-computed 16-bit values (i.e. patterns). As number of 1s grow, the computed pattern guarantees to spread the 1s uniformly such that the distance between two consecutive 1s is (close to) maximum (shown as Max 1-Dist in Table 1). Note that the set of possible patterns is known and pre-computed; however, where in the look-up table each pattern resides depends on the type of the decision function (linear, sub-linear, or super-linear).

3.2.3 Enforcing the New Decision. A new decision is loaded into the circular right shift register (shown in Figure 3) as an E-bit pattern. Each bit in this pattern corresponds to one (or more) block transfers. For every request, value of right-most bit in the pattern is checked. If 1, the current request will be transferred in DESC format, otherwise NRZ. The number of transfers each bit corresponds to indicates when the shift command of the circular right shift register will be triggered. At the finest *decision granularity*, each transfer can trigger the shift command, which means the decision pattern repeats after every E transfers. Note that the objective is to interleave DESC/NRZ(I) transfers as much as possible to avoid any undesirable latency overheads (as shown in Table 1).

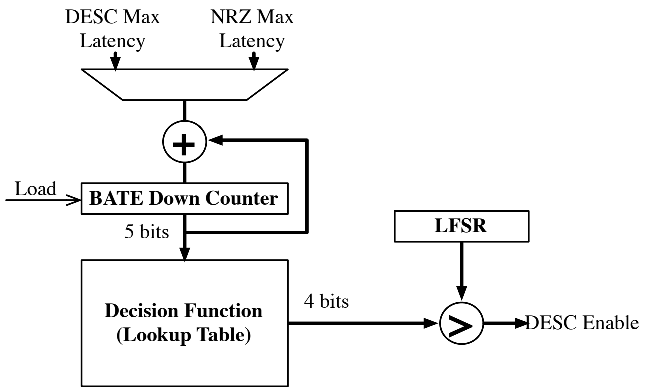

Figure 4: Illustrative example of the proposed architecture for burst adaptive time-based encoding.

In addition to PATE as a hybrid of DESC and NRZ encoding schemes, we also introduce PATE-I which is similar conceptually the same as PATE except that NRZI is used as the alternative encoding for when DESC is disabled.

3.3 BATE: Burst Adaptive Time-based Encoding

Phase adaptivity is practical when there is similarity between consecutive phases that could help make intelligent decision for incoming transfers. For the workloads with non-uniform behaviors, we propose Burst Adaptive Time-based Encoding (BATE), to shorten the reaction time to sudden burstiness in traffic and bring adaptivity to a very fine granularity—i.e., a single transfer. More precisely, BATE views a stream of requests $S = R1, R2, ..., Rs$ and looks at R_i to make a decision about encoding scheme for R_{i+1}. Like PATE, the decision function for BATE can be formulated using Equation 1.

For BATE encoding, x is the time budget left (i.e. maximum transfer time minus the elapsed time from the previous transfers). The key idea is that when the remaining time budget is low enough, the decision function has to safely recommend a DESC transfer for the next block. However, if budget is high (i.e. not enough time has passed since last DESC transfer), the function chooses NRZ for the next transfer. Otherwise, a random number r is generated and compared against the computed $f(x, n)$. If $f(x, n) > r$ then the decision is DESC, otherwise NRZ. Once a decision is made for the next transfer, the time budget is incremented by either α or β, depending on whether decision was DESC or NRZ, respectively.

Figure 4 demonstrates an example architecture for Burst Adaptive Time-based Encoding (BATE). The down-counter keeps track of the current time budget. To simplify the hardware implementation, decision function is implemented as a lookup table which contains quantized values of the BATE curve when the budget is in $[\beta, \alpha]$ range. The budget value is used to index the table and find an approximate (and pre-computed) decision function value. The found value is then compared against a pseudo-random number (i.e. *LFSR* block) and the output is used to enable or disable the DESC encoding.

For BATE implementation, we also consider an alternative where NRZ is replaced by NRZI, which is named BATE-I in our evaluations.

Table 2: Simulation parameters.

Component	Parameters
Processor core	four cores, 4-issue out-of-order core, 128 ROB entries, 3.2 GHz
IL1 cache (per core)	64KB, direct-mapped, 64B block, hit/miss delay 2/2
DL1 cache (per core)	64KB, 4-way, LRU, 64B block, hit/miss delay 2/2, MESI protocol
L2 cache (shared)	4MB, 16-way, LRU, 64B block, hit/miss delay 19/12
Temperature	350°K (77°C)
DRAM	2 DDR4-1600 memory channels, FP-FCFS

4 EXPERIMENTAL SETUP

In this section, the empirical setups for the tools and the characteristics of applications are explained. We assess the energy and delay potentials of the proposed adaptive time-based encoding when applied to a 4MB last level cache. We use hardware synthesis to evaluate the area, delay, and power overheads of the proposed architectures for ATE. A multicore systems is modeled to execution a mix of 12 parallel benchmark applications for evaluating the overall system energy and performance. Moreover, we implement three baseline systems including NRZ, NRZI, DESC, and Bus Invert Coding (BIC) [30] interfaces for comparison with ATE. We estimate the overall system power for ATE and each of the baseline interfaces using McPAT [22]. Similarly to the evaluations performed by the prior work on DESC [6], we explore the design space of each baseline interface to find the best performing configuration.

4.1 Hardware Synthesis

To assess the area, delay, and power overheads of proposed hardware, we use logic synthesis with FreePDK [12] at the 45nm technology node. Similarly to prior work [5], the results are then scaled down to 22nm. We use CACTI [24] to model all of the lookup tables required for the proposed microarchitectures.

4.2 System Architecture

We heavily modify the ESESC simulator [1] to model a multi-core out-of-order processor interfaced with two DDR4-1600 DRAM channels. The system includes a 4MB L2 cache comprising a total of 256 sub-arrays organized into 4 banks and 4 sub-banks per bank. Table 2 shows the simulation parameters.

4.3 Last-Level Cache Architecture

SRAM-based last level caches contribute to the energy and performance of microprocessors, significantly. As a result of consuming a large number of transistors for building a last level cache, excessive leakage energy is dissipated in its storage cells and peripheral circuitry. Numerous optimization techniques have been proposed in the literature to alleviate this problem [13, 15, 17, 18, 23, 28, 31], which have to be considered for designing an energy-efficient last level cache. A comprehensive search is required to find the most energy efficient cache architecture for a particular system. We employ CACTI and ESESC to perform a search in the design space of L2 cache and to find the most energy efficient cache organization and

Table 3: Applications and data sets.

Label	Benchmark	Suite	Input
CH	Cholesky	SPLASH-2	tk29.0
FFT	Fast Fourier Transform	SPLASH-2	1048576 data points
OC	Ocean	SPLASH-2	514x514 ocean
RDX	Radix	SPLASH-2	2M integers
RT	Ray Trace	SPLASH-2	car
IS	Integer Sort	NAS	Class A
MG	MG	NAS	Class A
HIS	Histogram	Phoenix	100MB file
LR	Linear Regression	Phoenix	50MB key file
PCA	Principal Component Analysis	Phoenix	1000 x 2000 matrix
SM	String Match	Phoenix	10MB key file
WC	Word Count	Phoenix	10MB text file

underlying technology for the cells and interconnects. Similarly to prior work [6], we explore various transistor types, including the ITRS high performance (HP), ITRS low power (LOP), and ITRS low standby power (LSTP) devices [24]. We also study the energy efficiency of LLC for various numbers of banks and data wires while simulating all 12 benchmark applications to find the most energy efficient cache design parameters for ATE and each of the baseline systems.

4.4 Benchmark Applications

We choose a mix of 12 data-intensive benchmark applications from three parallel suites: Phoenix [36], SPLASH-2 [35], and NAS [2]. All of the applications are compiled using GCC with the -O3 optimization flag. Table 3 shows the description of the evaluated benchmarks and their input sets. All of these parallel applications are simulated to completion.

5 EVALUATION

In this section, we first present the results of hardware synthesis for the proposed PATE(I) and BATE(I) architectures. Next, we provide system level energy and performance comparisons between the proposed architectures and the baseline systems to evaluate the impact of adaptive time-based encoding on energy-efficiency and performance. To better understand various optimization aspects of the proposed architectures, we present details of the decisions made for each benchmark application and a sensitivity analysis on various types of the decision function. We also illustrate the sensitivity of energy-efficiency to interval length in PATE(I).

5.1 Synthesis Results

Our synthesis results indicate negligible amounts of area, delay, and power overheads for both PAT(I) and BATE(I) controllers. The respective additional die areas consumed by the necessary hardware controllers for PATE(I) and BATE(I) are 312μm^2 and 268μm^2, which contribute to less than 1% of the 4MB LLC area. The peak power consumption of PATE(I) and BATE(I) controllers are 0.519W and 0.423W, respectively. Also, the synthesis results indicate that the decision making process for PATE(I) and BATE(I) controllers take up to 498ps and 426ps, respectively. We accurately consider these overheads in our simulations for performance and energy evaluations.

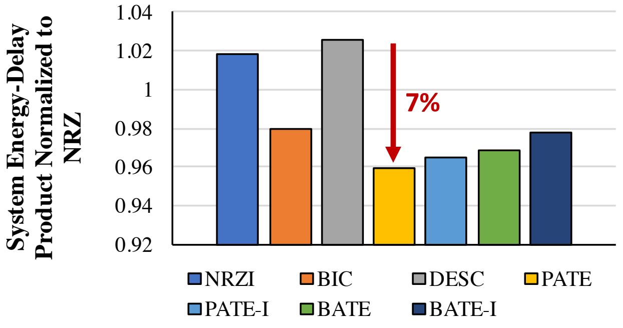

Figure 5: Energy efficiency of the proposed adaptive time-based encoding compared with the conventional binary NRZ(I), bus invert coding, and DESC. All the system energy-delay products are normalized to that of NRZ.

5.2 Energy Efficiency

Figure 5 illustrates the relative system energy-delay product for NRZ(I), bus invert coding, DESC, PATE(I) and BATE(I). The energy-delay products are computed as geometric means across of of the 12 benchmark applications. The results indicate that all variations of the proposed adaptive time-based encoding provide better energy efficiency compared to the baselines. PATE, however, achieves superior energy-efficiency among of the other schemes. As compared with DESC, PATE improves the overall system efficiency (*EDP*) by 7% on average. Please notice that DESC provides the worst energy efficiency due to significant increase in bandwidth consumption per block (recall the 3.15× latency increase in Figure 1). The superior energy-efficiency is achieved due to a reduction in average block transfer delay from 3.15× (in DESC) to 2.06×, while significant energy savings are still attained for PATE. This reduction in block transfer delay results in decreasing total execution time of the applications.

5.3 System Performance

Figure 6 shows the relative execution time for all 12 benchmark applications run on the NRZ(I), bus invert coding, DESC, PATE(I), and BATE(I) systems. As compared with DESC, the proposed adaptive encoding provides better execution times across all 12 applications. PATE achieves the most reduction in the average execution time (by 9%) and BATE-I, the least (7%). The most significant time reduction is achieved for the most cache intensive applications—e.g., 23% for Ocean. In summary, the proposed adaptive encoding reduce the overall execution time significantly; however, such performance improvement can only lead to improving system energy-efficiency if significant energy is also saved.

5.4 System Energy

Figure 7 shows the system energy consumption of the proposed adaptive encoding architectures and the baseline systems running the 12 benchmark applications. The results indicate that the proposed adaptive schemes are successful at utilizing time-based data encoding for saving system energy: the proposed architectures can achieve within 5% of the DESC system energy. As compared to the NRZ baseline, the proposed adaptive encoding schemes save 12-13% of the overall system energy, on average. In summary, the proposed adaptive encoding is able to decrease the overall system energy by reducing the LLC wire-flips across all 12 applications. Only for the application "Word Count", BATE consumes more energy than the baselines, which is mainly because of 1) increased leakage energy due to a longer execution time and 2) relatively less reduction in wire flips compared to other techniques.

5.5 Cache Design Space

The proposed adaptive time-based encoding creates new design opportunities for implementing more energy-efficient last level caches. Figure 8 shows the relative execution time and system energy influenced by the application of NRZ(I), bus invert coding, DESC, PATE(I) and BATE(I) to the last level cache. An ideal design point could be an architecture with minimal energy consumption and shortest execution time. As shown in the figure, the proposed adaptive time-based encoding provides new LLC design possibilities that are closer to the ideal system.

5.6 Encoding Decisions

The proposed adaptive time-based encoding is capable of striking a balance between consumed bandwidth and energy consumption. This is mainly accomplished through monitoring the LLC bandwidth and making decisions on whether a time-based code should be used for every transferred block. Our simulations indicate that decisions made by each mechanism depends on characteristics of the application and data set. Figure 9 illustrates the encoding decisions made by the proposed adaptive techniques for all of the benchmark applications. The results indicate that the adaptive mechanisms choose to employ time-based codes for 49-75% of all transferred blocks. Moreover, PATE(I) scheme employs time-based codes less often than BATE(I), which is due to monitoring the average bandwidth consumed by many more requests before making decisions.

5.6.1 Proportionality of Decisions. Figure 10 shows the sensitivity of system energy-delay product to the type of function used for decision making. The results indicate that a sub-linear function ($n = 2$) can better improve the overall energy-efficiency for all of the adaptive encoding mechanisms.

5.6.2 Length of PATE Intervals. We also observed that a medium sized interval ($L = 1024$) works better for the BATE(I) scheme (Figure 11).

6 RELATED WORK

Numerous optimization techniques for caches and on-chip communication have come to existence with different, yet orthogonal, objectives to what this paper presents. Solutions for optimizing access time and energy (e.g. [20]) or overall cache energy (e.g. cache compression [9, 10, 26, 29, 33] or cache bypassing [25]) have proven to be effective. However, they are orthogonal (and can complement) our solution of energy-efficient data exchange for cache interconnects. Our proposed time-based encoding mechanism improves cache energy efficiency by decreasing the number of data interconnects and their signal transition rate, which makes it distinguishable

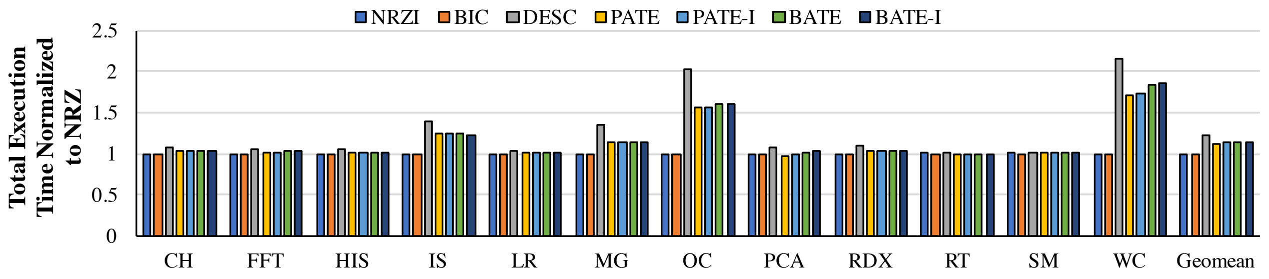

Figure 6: Total execution time of the benchmark applications running on the proposed architectures and the baseline systems.

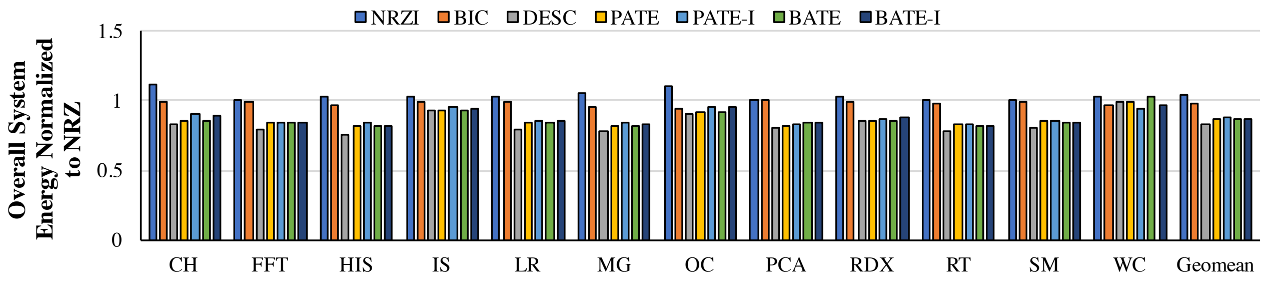

Figure 7: Overall system energy consumption for running the benchmark applications on the proposed architectures and the baseline systems.

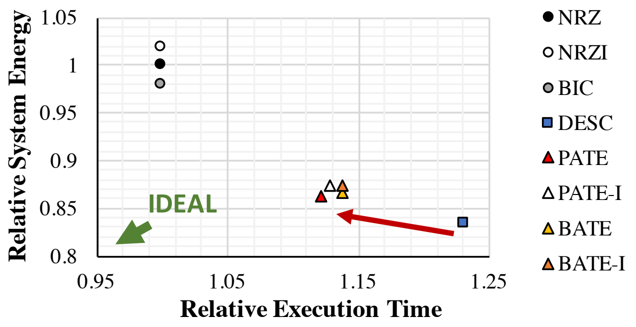

Figure 8: Design space of last level caches with conventional and the proposed encoding techniques.

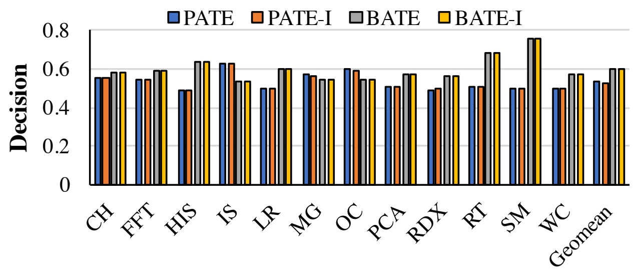

Figure 9: Encoding decisions made by the proposed adaptive mechanisms for the benchmark applications.

from (and applicable to) other proposals related to interconnect topology and wiring characteristics (e.g. [3, 24].

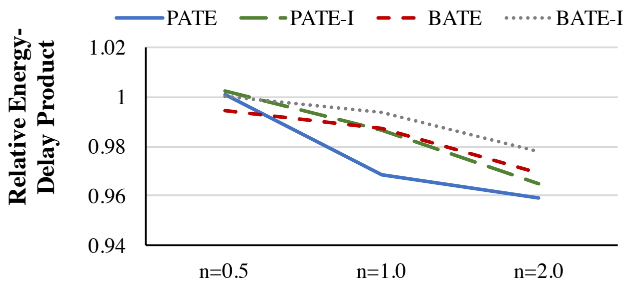

Figure 10: System energy-delay product vs. the proportionality of decisions.

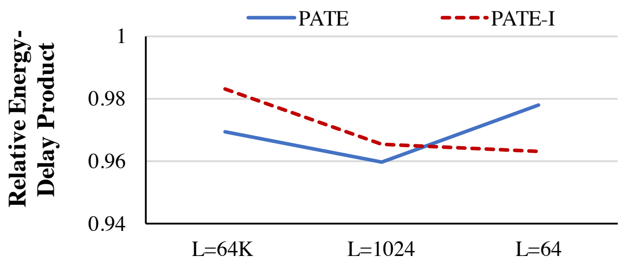

Figure 11: System energy-delay product as the interval length varies from 64 to $64K$ requests.

Related to dynamic energy and switching activity of the cache interconnects, different bus-encoding encoding techniques (e.g. bus-invert coding [30], on-line data clustering [34], etc) have shown success in bounding the number of state transitions over the wires. Our proposal shares the same objective and reduces the bit flips, but takes a complete different approach (i.e. time-base encoding) to represent information.

Besides dynamic, techniques for reducing static cache power (e.g. Cache Decay [19], Drowsy Cache [11] or others [21]) have been proposed to reduce cache leakage power. Although effective, these proposals are orthogonal to our solution and can be combined to achieve very highly energy-efficient cache designs.

Historically, information is represented in a time-based fashion for pulse position modulation (PPM) in telecommunications [27] and DESC [6] was the first proposal to adapt this concept in on-chip caches. However, DESC is very data sensitive while lacking adaptivity, and as shown in this paper, it can have huge negative influence on the sustainable bandwidth of the LLC interconnect. Our adaptive solution builds upon the DESC idea and finds a solution that makes the LLC responsive to the changes in the application; enables highly energy-efficient LLC designs.

7 CONCLUSIONS

Data movement on LLC interconnects contributes to a significant portion of energy in modern processors. This paper examined a novel adaptive time-based encoding to improve energy-efficiency of large last level caches. We observed that the proposed adaptation mechanism can improve the efficiency of large caches by creating new design opportunities with more energy-efficient interfaces.

ACKNOWLEDGMENTS

The authors would like to thank anonymous reviewers for useful feedback. This work was supported in part by University of Utah seed grant.

REFERENCES

[1] Ehsan K. Ardestani and Jose Renau. 2013. ESESC: A Fast Multicore Simulator Using Time-Based Sampling. In *International Symposium on High Performance Computer Architecture (HPCA)*. 448–459.

[2] D. H. Bailey et al. 1994. *NAS parallel benchmarks*. Technical Report. NASA Ames Research Center. Tech. Rep. RNR-94-007.

[3] Rajeev Balasubramonian, Naveen Muralimanohar, Karthik Ramani, and Venkatanand Venkatachalapathy. 2005. Microarchitectural Wire Management for Performance and Power in Partitioned Architectures. In *International Symposium on High-Performance Computer Architecture (HPCA)*. 28–39.

[4] Bradford M. Beckmann and David A. Wood. 2003. TLC: Transmission Line Caches. In *International Symposium on Microarchitecture (MICRO)*. 43–54.

[5] Mahdi Nazm Bojnordi and Engin Ipek. 2012. PARDIS: A programmable memory controller for the DDRx interfacing standards. In *International Symposium on Computer Architecture (ISCA)*. 13–24.

[6] Mahdi Nazm Bojnordi and Engin Ipek. 2013. DESC: Energy-efficient Data Exchange Using Synchronized Counters. In *International Symposium on Microarchitecture (MICRO)*. 234–246.

[7] Mahdi Nazm Bojnordi, Nariman Moezzi Madani, Mehdi Semsarzade, and Ali Afzali-Kusha. 2006. An Efficient Clocking Scheme for On-Chip Communications. In *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. 119–122.

[8] G. Chandra, P. Kapur, and K.C. Saraswat. 2002. Scaling trends for the on chip power dissipation. In *Interconnect Technology Conference (ITC)*. 170–172.

[9] Daniel Citron. 2004. Exploiting Low Entropy to Reduce Wire Delay. *IEEE Computer Architecture Letters* 3 (2004).

[10] Julien Dusser, Thomas Piquet, and André Seznec. 2008. *Zero-Content Augmented Caches*. Rapport de recherche RR-6705. INRIA.

[11] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge. 2002. Drowsy Caches: simple techniques for reducing leakage power. In *International Symposium on Computer Architecture*. 148–157.

[12] FreePDK [n. d.]. FreePDK 45nm Open-Access Based PDK for the 45nm Technology Node. ([n. d.]). http://www.eda.ncsu.edu/wiki/FreePDK.

[13] Masaki Fujigaya, Noriaki Sakamoto, Takao Koike, Takahiro Irita, Kohei Wakahara, Tsugio Matsuyama, Keiji Hasegawa, Toshiharu Saito, Akira Fukuda, Kaname Teranishi, Kazuki Fukuoka, Noriaki Maeda, Koji Nii, Takeshi Kataoka, and Toshihiro Hattori. 2013. A 28nm High-k metal-gate single-chip communications processor with 1.5GHz dual-core application processor and LTE/HSPA+-capable baseband processor. In *International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 156–157.

[14] Y. Fujiwara. 2013. Self-Synchronizing Pulse Position Modulation With Error Tolerance. *IEEE Transactions on Information Theory* 59, 9 (Sept 2013), 5352–5362.

[15] Varghese George, Sanjeev Jahagirdar, Chao Tong, K. Smits, Satish Damaraju, Scott Siers, Ves Naydenov, Tanveer Khondker, Sanjib Sarkar, and Puneet Singh. 2007. Penryn: 45-nm next generation Intel core 2 processor. In *IEEE Asia Solid-State Circuits Conference (ASSCC)*. 14–17.

[16] Nikos Hardavellas, Michael Ferdman, Anastasia Ailamaki, and Babak Falsafi. 2010. Power Scaling: the Ultimate Obstacle to 1K-Core Chips. In *Technical Report NWU-EECS-10-05*.

[17] D. James. 2012. Intel Ivy Bridge unveiled - The first commercial tri-gate, high-k, metal-gate CPU. In *IEEE Custom Integrated Circuits Conference (CICC)*. 1–4.

[18] E. Karl, Yih Wang, Yong-Gee Ng, Zheng Guo, F. Hamzaoglu, U. Bhattacharya, K. Zhang, K. Mistry, and M. Bohr. 2012. A 4.6GHz 162Mb SRAM design in 22nm tri-gate CMOS technology with integrated active VMIN-enhancing assist circuitry. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 230–232.

[19] S. Kaxiras, Z. Hu, and M. Martonosi. 2001. Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power. In *International Symposium on Computer Architecture*.

[20] Changkyu Kim, Doug Burger, and Stephen W. Keckler. 2002. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*. 211–222.

[21] Nam Sung Kim, David Blaauw, and Trevor Mudge. 2003. Leakage Power Optimization Techniques for Ultra Deep Sub-Micron Multi-Level Caches. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD'03)*. 627–.

[22] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. 2009. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *International Symposium on Computer Architecture*.

[23] N. Maeda, S. Komatsu, M. Morimoto, and Y. Shimazaki. 2012. A 0.41 uA standby leakage 32Kb embedded SRAM with Low-Voltage resume-standby utilizing all digital current comparator in 28nm HKMG CMOS. In *Symposium on VLSI Circuits (VLSIC)*. 58–59.

[24] N. Muralimanohar, R. Balasubramonian, and N. Jouppi. 2007. Optimizing NUCA Organizations and Wiring Alternatives for Large Caches With CACTI 6.0. In *International Symposium on Microarchitecture (MICRO)*.

[25] J. J. K. Park, Y. Park, and S. Mahlke. 2016. A bypass first policy for energy-efficient last level caches. In *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*. 63–70.

[26] G. Pekhimenko, E. Bolotin, N. Vijaykumar, O. Mutlu, T. C. Mowry, and S. W. Keckler. 2016. A case for toggle-aware compression for GPU systems. In *International Symposium on High Performance Computer Architecture (HPCA)*. 188–200.

[27] J. G. Proakis. 1995. *Digital Communications*. Third Edition, McGraw-Hill.

[28] S. Rusu, S. Tam, H. Muljono, D. Ayers, and J. Chang. 2006. A Dual-Core Multi-Threaded Xeon Processor with 16MB L3 Cache. In *IEEE International Solid-State Circuits Conference (ISSCC). Digest of Technical Papers*. 315–324.

[29] A. Seznec. 1994. Decoupled sectored caches: conciliating low tag implementation cost. In *International Symposium on Computer architecture (ISCA)*. 384–393.

[30] Mircea R. Stan and Wayne P. Burleson. 1995. Bus-invert coding for low-power i/o. *IEEE Transactions on VLSI Systems* (1995), 49–58.

[31] Fumihiko Tachibana, Osamu Hirabayashi, Yasuhisa Takeyama, Miyako Shizuno, Atsushi Kawasumi, Keiichi Kushida, Azuma Suzuki, Yusuke Niki, Shinichi Sasaki, Tomoaki Yabe, and Yasuo Unekawa. 2013. A 27% active and 85% standby power reduction in dual-power-supply SRAM using BL power calculator and digitally controllable retention circuit. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 320–321.

[32] A.N. Udipi, N. Muralimanohar, and R. Balasubramonian. 2009. Non-uniform power access in large caches with low-swing wires. In *International Conference on High Performance Computing (HiPC)*. 59–68.

[33] Luis Villa, Michael Zhang, and Krste Asanovic. 2000. Dynamic zero compression for cache energy reduction. In *MICRO*. 214–220.

[34] S. Wang and E. Ipek. 2016. Reducing data movement energy via online data clustering and encoding. In *International Symposium on Microarchitecture (MICRO)*. 1–13.

[35] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. 1995. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *ISCA*. 24–36.

[36] Richard M. Yoo, Anthony Romano, and Christos Kozyrakis. 2009. Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system. In *International Symposium on Workload Characterization*.

[37] Hui Zhang and J. Rabaey. 1998. Low-swing interconnect interface circuits. In *International Symposium on Low Power Electronics and Design*. 161 –166.